D. patenting ???

L4: Entry 1 of 2 File: USPT Sep 18, 2001

09322137

US-PAT-NO: 6292932
DOCUMENT-IDENTIFIER: US 6292932 B1

TITLE: System and method for converting from one modeling language to another

DATE-ISSUED: September 18, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Baisley; Donald Edward	Laguna Hills	CA		
Iyengar; Sridhar Srinivasa	Irvine	CA		
Sawhney; Ashit	Lake Forest	CA		

US-CL-CURRENT: 717/114; 707/100, 717/138

CLAIMS:

What is claimed is:

1. In a computing system having a repository program being executed by said system and a means for storing data, a method for converting a UML model to a MOF model within said repository, said method comprising:

a. selecting a package within said UML model, hereafter UML package, to be exported to said MOF model;

b. exporting said UML package and its elements to said MOF model;

c. recursively setting relations between MOF objects of said UML package that correspond to relations between UML objects in said package; and,

d. creating MOF reference objects for navigable MOF association ends.

2. The method as in claim 1, wherein said step of exporting said UML package further comprises:

for each element owned by said UML package, perform the following steps:

1) creating a MOF package object;

2) determining whether or not said UML package includes an element, and if so;

3) determining whether or not said element is a package, and if not;

4) determining whether or not said element is a class, and if not;

5) determining whether or not said element is an association, and if so;

6) creating a MOF association and a MOF association end for each UML association end.

3. The method as in claim 2, where it is determined in step 3) thereof that said element is a package, repeating all of the steps of claim 2 for exporting the contained package.

4. The method as in claim 2, where it is determined in step 5 thereof that said element is not an association, further comprising the steps of:

a. determining whether or not said element is an exception, and if not;

b. repeating all of the steps of claim 2.

5. The method as in claim 4, where it is determined that said element is not an association, further comprising the step of creating a MOF exception.

6. The method as in claim 1, where it is determined that said element is a class, further comprising the steps of:

a. determining whether or not class stereotype is defined, and if not;

b. creating a MOF class object; and

c. for each UML attribute creating a MOF attribute object.

7. The method as in claim 6 further including the steps of:

a. creating a MOF operation object; and,

b. creating a MOF parameter for each UML parameter of the operation.

8. The method as in claim 6 further including the steps of:

a. creating a MOF exception object; and,

b. creating a MOF parameter for each UML parameter of the exception.

9. The method as in claim 2 wherein said step of determining whether or not said element is an association further includes:

a. determining whether or not said MOF element is a navigable association end, and if so;

b. determining if said UML package owning said association also the owner of the class that is the type of the opposite association end, and if so;

c. creating a MOF reference in said class with name and type matching said association end.

10. The method as in claim 9 where in step a thereof it is determined that said MOF element is not a navigable association end, further comprising the steps of:

a. determining whether or not said MOF element is an operation, and if so;

b. setting exceptions that can be raised by the MOF operation to MOF exceptions created for UML exceptions that can be raised by the corresponding UML operation.

11. The method as in claim 1, where it is determined that said element is a class and stereotype is defined, further comprising the steps of:

a. determining if stereotype is a datatype, and if so;

b. creating a MOF data type object.

12. The method as in claim 1, where it is determined that said element is a class and stereotype is defined, further comprising the steps of:

a. determining if stereotype is an enumeration, and if so;

b. setting type code to enumeration type using UML attribute names as enumeration literals.

13. The method as in claim 1, where it is determined that said element is a class and stereotype is defined, further comprising the steps of:

a. determining if stereotype is a structure, and if so;

b. determining if said class has a supertype, and if so;

c. determining if said class has attributes, and if so;

d. setting typecode to structured type having members based on UML attributes plus inherited UML attributes.

14. In a computing system having a repository program being executed by said system and a means for storing data, a system for converting a UML model to a MOF model within said repository, said system comprising:

a. means for selecting a package within said UML model, hereafter UML package, to be exported to said MOF model;

b. means for exporting said UML package and its elements to said MOF model;

c. means for recursively setting relations between MOF objects of said UML package that correspond to relations between UML objects in said package; and,

d. means for creating MOF reference objects for navigable MOF association ends.

15. A storage medium encoded with machine-readable computer program code for converting a UML model to a MOF model, wherein, when said computer program code is executed by a computer, said computer performs the steps of:

a. selecting a package within said UML model, hereafter UML package, to be exported to said MOF model;

b. exporting said UML package and its elements to said MOF model;

c. recursively setting relations between MOF objects of said UML package that correspond to relations between UML objects in said package; and,

d. creating MOF reference objects for navigable MOF association ends.

16. The storage medium as in claim 15, wherein said step of exporting said UML package further comprises:

for each element owned by said UML package, perform the following steps:

1) creating a MOF package object;

2) determining whether or not said UML package includes an element, and if so;

3) determining whether or not said element is a package, and if not;

4) determining whether or not said element is a class, and if not;

5) determining whether or not said element is an association, and if so;

6) creating a MOF association and a MOF association end for each UML association end.

17. The storage medium as in claim 16, where it is determined in step 3) thereof that said element is a package, repeating all of the steps of claim 16 for exporting the contained package.

18. The storage medium as in claim 16, where it is determined in step 5 thereof that said element is not an association, further comprising the steps of:

a. determining whether or not said element is an exception, and if not;

b. repeating all of the steps of claim 2.

19. The storage medium as in claim 18, where it is determined that said element is not an association, further comprising the step of creating a MOF exception.

20. The storage medium as in claim 16 wherein said step of determining whether or not said element is an association further includes:

a. determining whether or not said MOF element is a navigable association end, and if so;

b. determining if said UML package owning said association also the owner of the class that is the type of the opposite association end, and if so;

c. creating a MOF reference in said class with name and type matching said association end.

21. The storage medium as in claim 15, where it is determined that said element is a class, further comprising the steps of:

a. determining whether or not class stereotype is defined, and if not;

b. creating a MOF class object; and

c. for each UML attribute creating a MOF attribute object.

22. The storage medium as in claim 21 further including the steps of:

a. creating a MOF operation object; and,

b. creating a MOF parameter for each UML parameter of the operation.

23. The storage medium as in claim 21 further including the steps of:

a. creating a MOF exception object; and,

b. creating a MOF parameter for each UML parameter of the exception.

24. The storage medium as in claim 22 where in step a thereof it is determined that said MOF element is not a navigable association end, further comprising the steps of:

a. determining whether or not said MOF element is an operation, and if so;

b. setting exceptions that can be raised by the MOF operation to MOF exceptions created for UML exceptions that can be raised by the corresponding UML operation.

25. The storage medium as in claim 15, where it is determined that said element is a class and stereotype is defined, further comprising the steps of:

a. creating a MOF data type object; and,

b. setting type code to enumeration type using UML attribute names as enumeration literals.

First Hit Fwd Refs

Generate Collection Print

L3: Entry 1 of 2 File: USPT Apr 30, 2002

US-PAT-NO: 6381743
DOCUMENT-IDENTIFIER: US 6381743 B1

09282230

TITLE: Method and system for generating a hierarchial document type definition for data interchange among software tools

DATE-ISSUED: April 30, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mutschler, III; Eugene Otto	San Clemente	CA		

US-CL-CURRENT: 717/104; 707/104.1, 717/101, 717/108, 717/116, 717/137

CLAIMS:

What is claimed is:

1. In a software development framework having a repository and one or more software systems wherein said repository contains a meta-model and said software systems, which store instances of said meta-model, a method for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, said method comprising the steps of:

a. extracting a fixed component and a variable component of said meta-model;

b. parsing said variable component into a first set of constituent components for a first instance of said variable component;

c. extracting a list of repeated components from said first set of constituent components;

d. determining the hierarchical order and inheritance of said repeated components in said list of repeated components;

e. transforming said repeated components in said list of repeated components into corresponding components of a generalized software language;

f. transforming each of said first set of constituent components into corresponding components of said generalized software language;

g. transforming said first instance of variable component into corresponding components of said generalized software language;

h. repeating steps b to g for a second instance of said variable component;

i. transforming said fixed component into corresponding components of said generalized software language;

j. distributing said corresponding components to said second instance of said software model; and,

k. using said distributed said corresponding components to control the syntax of said generalized data transfer language to exchange said meta-model instances.

2. A method as in claim 1 wherein step d thereof further includes the step of transforming more than one of said repeated components into a single corresponding component.

3. The method as in claim 2 wherein said single corresponding component is an XML entity definition.

4. The method as in claim 1 wherein said meta-model is the Universal Modeling Language (UML).

5. The method as in claim 1 wherein said data transfer language is extensible Markup Language (XML).

6. The method as in claim 1 wherein said generalized software language is the Document Type Definition (DTD) specification language for XML.

7. The method as in claim 1 wherein said software systems are software modeling tools.

8. A method as in claim 1 wherein said meta-model is a meta-metamodel with instances that are themselves meta-models.

9. A method as in claim 1 further including facilitating exchange of said software model data between two software systems.

10. A storage medium for use in a software development framework having a repository and at least two software systems wherein said repository contains a meta-model and said software systems, which store instances of said meta-model, said medium encoded with machine-readable computer program code for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, wherein when the computer program code is executed by a computer, the computer performs the steps of:

a. extracting a fixed component and a variable component of said meta-model;

b. parsing said variable component into a first set of constituent components for a first instance of said variable component;

c. extracting a list of repeated components from said first set of constituent components;

d. determining the hierarchical order and inheritance of said repeated components in said list of repeated components;

e. transforming said repeated components in said list of repeated components into corresponding components of a generalized software language;

f. transforming each of said first set of constituent components into corresponding components of said generalized software language;

g. transforming said first instance of variable component into corresponding components of said generalized software language;

h. repeating steps b to g for a second instance of said variable component;

j. transforming said fixed component into corresponding components of said generalized software language;

l. distributing said corresponding components to said second instance of said software model; and,

m. using said distributed said corresponding components to control the syntax of said generalized data transfer language to exchange said meta-model instances.

11. A medium as in claim 10 wherein step d thereof further includes the step of transforming more than one of said repeated components into a single corresponding component.

12. The medium as in claim 11 wherein said single corresponding component is an XML entity definition.

13. The medium as in claim 10 wherein said meta-model is the Unified Modeling Language (UML).

14. The medium as in claim 10 wherein said data transfer language is extensible Markup Language (XML).

15. The medium as in claim 10 wherein said generalized software language is the Document Type Definition (DTD) specification language for XML.

16. The medium as in claim 10 wherein said software systems are software modeling tools.

17. A medium as in claim 10 wherein one of said instances of said meta-model is based on a software model.

18. A method as in claim 10 further including facilitating exchange of said software model data between two software systems.

19. In a software development framework having a repository and at least two software systems wherein said repository contains a meta-model and said software systems, which store instances of said meta-model, a method for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, said method comprising the steps of:

a. mapping primary objects of said meta-model to constructs of said generalized data transfer language;

b. mapping component parts and relationships of said primary meta-model objects to component constructs of said generalized data transfer language;

c. further mapping of groups of said relationships of said meta-model to aggregate constructs of said data transfer language;

d. mapping grouping mechanisms of said meta-model to grouping constructs of said generalized data transfer language;

e. defining algorithms for traversing said meta-model to obtain information about said component parts and relationships of said primary components necessary for preserving said information in a process of transforming said component parts and relationships, primary objects and grouping mechanisms into generalized software language components used to express said constructs, component constructs and grouping constructs of said generalized data transfer language;

f. defining algorithms for traversing said meta-model to obtain information about said groups of said relationships for the purpose of transforming said groups of said relationships into aggregate constructs of said generalized software language used to express said aggregate constructs of said generalized data transfer language;

g. defining algorithms for transforming groups of said relationships into generalized software language constructs used to express said aggregate constructs of said generalized data transfer language; and,

h. expressing relationships among said generalized software language components, whereby reliable and correct programs to perform said transforming of said component parts and relationships, primary objects and grouping mechanisms into said generalized software language components can be written.

First Hit Fwd Refs

Generate Collection Print

L2: Entry 1 of 2 File: USPT Jun 26, 2001

09282102

US-PAT-NO: 6253366
DOCUMENT-IDENTIFIER: US 6253366 B1

TITLE: Method and system for generating a compact document type definition for data interchange among software tools

DATE-ISSUED: June 26, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mutschler, III; Eugene Otto	San Clemente	CA		

US-CL-CURRENT: 717/104; 707/100, 717/107, 717/114

CLAIMS:

What is claimed is:

1. In a software development framework having a repository and one or more software systems wherein said repository contains a meta-model and said software systems, which store instances of said meta-model, a method for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, said method comprising the steps of:

a. extracting a fixed component and a variable component of said meta-model;

b. parsing said variable component into a first set of constituent components for a first instance of said variable component;

c. extracting a list of repeated components from said first set of constituent components;

d. transforming said repeated components in said list of repeated components into corresponding components of a generalized software language;

e. transforming each of said first set of constituent components into corresponding components of said generalized software language;

f. transforming said first instance of said variable component into corresponding components of said generalized software language;

g. repeating steps b through f for a second instance of said variable component;

h. transforming said fixed component into corresponding components of said generalized software

language;

i. distributing said corresponding components to another instance of said software model; and,

j. using said corresponding components distributed in the preceding step to control the syntax of said generalized data transfer language for exchanging said meta-model instances.

2. A method according to claim 1 wherein step c thereof further includes the step of generating separate objects for each list of repeated components and using said separate objects to transform multiple instances of said repeated components into corresponding components of said generalized software language.

3. The method as in claim 1 wherein each of said generalized software language components created by said transformation of each of said repeated components in said list of repeated components is an XML entity definition.

4. The method as in claim 1 wherein said generalized software language components created by said transformation of said constituent components are XML element definitions.

5. The method as in claim 1 wherein said generalized software language components created by said transformation of said instances of said variable components are XML element definitions.

6. A method according to claim 1 wherein step f thereof further includes the step of incorporating references to said corresponding components of said general purpose software language created by said transformation of each of said repeated components in said list of repeated components.

7. A method according to claim 1 wherein said meta-model is a meta-metamodel with instances that are themselves meta-models.

8. A method according to claim 1 further including facilitating exchange of data of said software model between two software systems.

9. A method as in claim 1 wherein said meta-model is the Unified Modeling Language (UML).

10. The method in claim 1 wherein said data transfer language is extensible Markup Language (XML).

11. The method as in claim 1 wherein said generalized data transfer language is the Document Type Definition (DTD) specification language for XML.

12. The method as in claim 1 wherein said software systems are software modeling tools.

13. A storage medium for use in a software development framework having a repository and at least two software systems wherein said repository contains a meta-model and said software systems, which are capable of storing instances of said meta-model, said medium encoded with machine-readable computer program code for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, wherein when the computer program code is executed by a computer, the computer performs the steps of:

a. extracting a fixed component and a variable component of said meta-model;

b. parsing said variable component into a first set of constituent components for a first instance of said variable component;

c. extracting a list of repeated components from said first set of constituent components;

d. transforming said repeated components in said list of repeated components into corresponding components of a generalized software language;

e. transforming each of said first set of constituent components into corresponding components of said generalized software language;

f. transforming said first instance of variable component into corresponding components of said generalized software language;

g. repeating steps b to f for a second instance of said variable component;

h. transforming said fixed component into corresponding components of said generalized software language;

i. distributing said corresponding components to said second instance of said software model; and,

j. using said distributed said corresponding components to control the syntax of said generalized data transfer language to exchange said meta-model instances.

14. A medium according to claim 13 wherein step c thereof further includes the step of generating separate objects for each list of repeated components and using said separate objects to transform multiple instances of said repeated components into corresponding components of said generalized software language.

15. A medium according to claim 13 wherein said instance of said meta-model is based on a software model.

16. A medium according to claim 13 further including facilitating exchange of data of said software model between two software systems.

17. A medium according to claim 13 wherein said meta-model is the Unified Modeling Language (UML).

18. A medium according to claim 13 wherein said data transfer language is extensible Markup Language (XML).

19. A medium according to claim 13 wherein said generalized software language is the Document Type Definition (DTD) specification for XML.

20. A medium according to claim 13 wherein said software systems are software modeling tools.

21. A medium according to claim 13 wherein one of said instances of said meta-model is based on

a software model.

22. A medium according to claim 13 further including facilitating exchange of said software model data between two software systems.

23. In a software development framework having a repository and at least two software systems wherein said repository contains a meta-model and said software systems, which store instances of said meta-model, a method for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, said method comprising the steps of:

a. mapping primary objects of said meta-model to constructs of said generalized data transfer language;

b. mapping component parts and relationships of said primary meta-model objects to component constructs of said generalized data transfer language;

c. further mapping of groups of said relationships of said meta-model to aggregate constructs of said data transfer language;

d. mapping grouping mechanisms of said meta-model to grouping constructs of said generalized data transfer language;

e. defining algorithms for traversing said meta-model to obtain information about said component parts and relationships of said primary components necessary for preserving said information in a process of transforming said component parts and relationships, primary objects and grouping mechanisms into generalized software language components used to express said constructs, component constructs and grouping constructs of said generalized data transfer language;

f. defining algorithms for transforming groups of said relationships into generalized software language constructs used to express said aggregate constructs of said generalized data transfer language; and,

g. expressing relationships among said generalized software language components, whereby reliable and correct programs to perform said transforming of said component parts and relationships, primary objects and grouping mechanisms into said generalized software language components can be written.

First Hit Fwd Refs

Generate Collection Print

L1: Entry 1 of 2 File: USPT Sep 11, 2001

09282345

US-PAT-NO: 6289501
DOCUMENT-IDENTIFIER: US 6289501 B1

TITLE: Method for generating simple document type definitions

DATE-ISSUED: September 11, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mutschler, III; Eugene Otto	San Clemente	CA		

US-CL-CURRENT: 717/114; 715/513

CLAIMS:

What is claimed is:

1. In a software development framework having repository and at least two software systems wherein said repository contains a meta-model and said software systems, which store instances of said meta-model, a method for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, said method comprising the steps of:

a. extracting a fixed component and a variable component of said meta-model;

b. parsing said variable component into constituent components;

c. transforming each of said constituent components into corresponding component of a generalized software language;

d. repeating steps b and c for each instance of said variable component;

e. transforming each instance of said variable component into corresponding components of said generalized software language;

f. transforming said fixed components into corresponding components of said generalized software language;

g. distributing said corresponding components to another one of said software systems; and,

h. using said distributed corresponding components to control the syntax of said generalized data transfer language to exchange said meta-model instances.

2. The method as in claim 1 wherein said meta-model is the Unified Modeling Language (UML).

3. The method in claim 1 wherein said data transfer language is eXtensible Markup Language (XML).

4. The method as in claim 1 wherein said generalized software language is the Documeut Type Definition (DTD) specification language for XML.

5. The method as in claim 1 wherein said software systems are software modeling tools.

6. A method according to claim 1 wherein said meta-model is a meta-metamodel with instances that are themselves meta-models.

7. A method as in claim 1 further including facilitating exchange of said software model data between two software systems.

8. A storage medium for use in a software developement framework having a repository and at least two software systems wherein said repository contains a meta-model and said software system, which store instances of said meta-model, said medium encoded with machine-readable computer program code for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, wherein when the computer program code is executed by a computer, the computer performs the steps of:

a. extracting a fixed component and a variable component of said meta-model;

b. parsing said variable component into constituent components;

c. transforming each of said constituent components into corresponding components of a generalized software language;

d. repeating steps b and c for each instance of said variable component;

e. transforming each instance of said variable component into corresponding components of said generalized software language;

f. transforming said fixed components into corresponding components of said generalized software language;

g. distributing said corresponding components to another one of said software systems; and,

h. using said distributed corresponding components to control the syntax of said generalized data transfer language to exchange said meta-model instances.

9. The medium as in claim 8 wherein said meta-model is the Unified Modeling Language (UML).

10. The medium as in claim 8 wherein said generalized data transfer language is eXtensible Markup Language (XML).

11. The medium as in claim 8 wherein said generalized software language is the Document Type Definition (DTD) specification language for XML.

12. The medium as in claim 8 wherein said software systems are software modeling tools.

13. A medium according to claim 8 wherein said meta-model is a meta-metamodel with instances that are themselves meta-models.

14. A medium as in claim 8 further including facilitating exchange of said software model data between two software systems.

15. In a software development framework having a repository and at least two software systems wherein said repository contains a meta-model and said software systems, which store instances of said meta-model, a method for enabling exchange of said instances of said meta-model among said software systems using a generalized data transfer language, said method comprising the steps of:

a. mapping primary objects of said meta-model to constructs of said generalized data transfer language;

b. mapping component parts and relationships of said primary meta-model objects to component constructs of said generalized data transfer language;

c. mapping grouping mechanisms of said meta-model to grouping constructs of said generalized data transfer language;

d. defining algorithms for traversing said meta-model to obtain information about said component parts and relationships of said primary components necessary for preserving said information in a process of transforming said component parts and relationships, primary objects and grouping mechanism into generalized software language contents used to express said constructs, component constructs and grouping constructs of said generalized data transfer language; and,

e. expressing relationships among said generalized software language components, whereby reliable and correct programs to perform said transforming of said component parts and relationships, primary objects and grouping mechanisms into said generalized software language components can be written.